

Capítulo 3.

La predicción de beneficios del mercado bursátil

Este segundo caso de estudio va más allá en el uso de técnicas de minería de datos. El dominio específico utilizado para ilustrar estos problemas es el de los sistemas automáticos de comercio de acciones. Se va a abordar la tarea de construir un sistema de comercio de valores basados en los modelos de predicción obtenidos con los datos de todos los días las cotizaciones de bolsa. Varios modelos serán juzgados con el objetivo de predecir el rendimiento futuro del índice del mercado S & P 500. Estas predicciones se utilizan junto con una estrategia de negociación para llegar a una decisión con respecto a las órdenes del mercado para generar. En este capítulo se trata de varios temas nuevos de minería de datos, entre los que son (1) el uso de R para analizar los datos almacenados en una base de datos, (2) cómo manejar los problemas de predicción, con un tiempo de orden entre las observaciones de datos (también conocido como de series de tiempo) y (3) un ejemplo de las dificultades de traducir las predicciones del modelo en decisiones y acciones en aplicaciones del mundo real.

3.1. Descripción del problema y objetivos.

La existencia de una enorme cantidad de data histórica sugiere que la minería de datos puede ofrecer una ventaja competitiva sobre la inspección humana de estos datos.

La meta general de la negociación de valores es mantener un portafolio de bienes basados en comprar y vender órdenes. El objetivo a largo plazo es alcanzar tantas ganancias como sea posible generadas a través del comercio de acciones. Se tratará de maximizar las ganancias durante un futuro periodo de prueba, por medio del comercio de acciones (comprar, vender, mantener). La estrategia usará como base para la toma de decisión las indicaciones arrojadas por el resultado del proceso de minería de datos. El principal criterio de evaluación serán los resultados operacionales de aplicar el conocimiento descubierto por el proceso de minería de datos.

3.2. Datos disponibles.

Los datos están disponibles en la página:

<http://www.liaad.up.pt/~ltorgo/DataMiningWithR>.

Con el fin de obtener los datos, se procede con las siguientes instrucciones:

```
> library(DMwR)
> data(GSPC)
```

La primera instrucción solo se requiere si no se ha iniciado sesión en R anteriormente. La segunda instrucción cargará un objeto, GSPC, de la clase xts. En la página del libro se

puede encontrar dos formatos alternativos. El primero es por valores separados por coma (CSV), el otro formato es una base de datos en MySQL. Posteriormente se ilustrará como cargar los datos en R mediante estas dos alternativas.

Independientemente de la fuente que se utilice, los datos de acciones diarias incluye información respecto a las siguientes propiedades:

- Fecha de la sesión bursátil.
- Precio de apertura al comienzo de la sesión.
- Precio más alto durante la sesión.
- Precio más bajo.
- Precio de cierre de la sesión.
- Volumen de transacciones.
- Precio ajustado de cierre.

3.3.2. Leyendo los datos desde un archivo CSV.

Como se mencionó anteriormente, en la página web del libro se pueden encontrar diferentes fuentes que contienen los datos para utilizar en este caso de estudio. Si se decide utilizar el archivo CSV, se descargará un archivo cuyas primeras líneas lucirán de la siguiente manera:

"Index"	"Open"	"High"	"Low"	"Close"	"Volume"	"AdjClose"
1970-01-02	92.06	93.54	91.79	93	8050000	93
1970-01-05	93	94.25	92.53	93.46	11490000	93.46
1970-01-06	93.46	93.81	92.13	92.82	11460000	92.82
1970-01-07	92.82	93.38	91.93	92.63	10010000	92.63
1970-01-08	92.63	93.47	91.99	92.68	10670000	92.68
1970-01-09	92.68	93.25	91.82	92.4	9380000	92.4
1970-01-12	92.4	92.67	91.2	91.7	8900000	91.7

Asumiendo que se ha descargado el archivo, y se ha guardado bajo el nombre de "sp500.csv" en el directorio de trabajo actual de la sesión de R, se puede cargar en R y crear un objeto xts con los datos, con el siguiente comando:

```
> GSPC <- as.xts(read.zoo("sp500.csv", header = T))
```

La **función read.zoo()** del paquete zoo lee un archivo CSV y los transforma los datos en un objeto zoo asumiendo que la primera columna contiene las etiquetas de tiempo. La **función as.xts()** coacciona el objeto resultante en un objeto de la clase xts.

3.3.3. Obteniendo los datos de la Web.

Otra alternativa de obtener los datos es utilizar el servicio gratis suministrado por Yahoo finanzas, el cual nos permite descargar un archivo CSV con los datos que se desea. El paquete de R **tseries**

incluye la **función get.hist.quote()** que se puede utilizar descargando los datos en un objeto zoo. Lo siguiente es un ejemplo del uso de esta función para obtener los datos de S & P 500:

```
> library(tseries)
> GSPC <- as.xts(get.hist.quote("^GSPC",start="1970-01-02",
  quote=c("Open", "High", "Low", "Close","Volume","AdjClose")))
...
...

> head(GSPC)

      Open  High   Low Close  Volume AdjClose
1970-01-02 92.06 93.54 91.79 93.00  8050000    93.00
1970-01-05 93.00 94.25 92.53 93.46 11490000    93.46
1970-01-06 93.46 93.81 92.13 92.82 11460000    92.82
1970-01-07 92.82 93.38 91.93 92.63 10010000    92.63
1970-01-08 92.63 93.47 91.99 92.68 10670000    92.68
1970-01-09 92.68 93.25 91.82 92.40  9380000    92.40

> GSPC <- as.xts(get.hist.quote("^GSPC",
  start="1970-01-02",end='2009-09-15',
  quote=c("Open", "High", "Low", "Close","Volume","AdjClose")))
```

Otra manera de obtener los datos desde la web (pero no la única, como se verá posteriormente), es mediante el uso de la **función getSymbols()** del paquete quantmod. Nuevamente, este es un paquete extra que se deberá instalar con anterioridad. La **función getSymbols()** en conjunto con otras funciones del paquete, proveen una manera simple pero poderosa de obtener los datos de diferentes fuentes. A continuación se muestra algunos ejemplos:

```
> library(quantmod)
> getSymbols("^GSPC")

> getSymbols("^GSPC", from = "1970-01-01", to = "2009-09-15")
> colnames(GSPC) <- c("Open", "High", "Low", "Close", "Volume",
+   "AdjClose")
```

Se puede tener de hecho, varios símbolos asociados con diferentes fuentes de datos, cada uno con sus propios parámetros. Todo esto se puede especificar al comenzar la sesión de R con la **función setSymbolLookup()**, como se puede ver en el siguiente ejemplo:

```

> setSymbolLookup(IBM=list(name='IBM',src='yahoo'),
+                 USDEUR=list(name='USD/EUR',src='oanda'))
> getSymbols(c('IBM','USDEUR'))

> head(IBM)
              IBM.Open  IBM.High  IBM.Low  IBM.Close  IBM.Volume  IBM.Adjusted
2007-01-03      97.18     98.40    96.26     97.27     9196800         92.01
2007-01-04      97.25     98.79    96.88     98.31    10524500         93.00
2007-01-05      97.60     97.95    96.91     97.42     7221300         92.16
2007-01-08      98.50     99.50    98.35     98.90    10340000         93.56
2007-01-09      99.08    100.33    99.07    100.07    11108200         94.66
2007-01-10      98.50     99.05    97.93     98.89     8744800         93.55

> head(USDEUR)
              USDEUR
2009-01-01  0.7123
2009-01-02  0.7159
2009-01-03  0.7183
2009-01-04  0.7187
2009-01-05  0.7188
2009-01-06  0.7271

```

En estos códigos se han especificado varias formas de obtener los datos desde la web de dos diferentes símbolos: IBM de Yahoo! Finanzas, y la tasa de cambio US Dollar – Euro de Oanda. Esto fue hecho a través de la **función setSymbolLookup()**, la cual se asegura de que cualquier uso subsecuente de la **función getSymbols()** en la sesión actual de R con los identificadores especificados en la llamada, usaran los ajustes que quieran.

3.2.4. Leyendo los datos de una base de datos MySQL.

Otra alternativa de almacenar los datos utilizados en este caso de estudio es en una base de datos MySQL. En la página web del libro hay un archivo SQL que se puede descargar y ejecutar dentro de MySQL para cargar los ajustes en una tabla de la base de datos. Después de crear la base de datos, se ejecuta el archivo SQL. Asumiendo que el archivo está en el mismo directorio desde donde se ha introducido MySQL, y que la base de datos que se ha creado es llamada **Quotes**, uno se puede conectar a MySQL y tipear:

```

mysql> use Quotes;
mysql> source sp500.sql;

```

La SQL **statements** contenido en el archivo “sp500.sql” (el archivo descargado desde la página web del libro) creará una tabla llamada “gspc” e insertará varios registros en esta tabla, conteniendo la data disponible para este caso de estudio. Lo dicho anteriormente se puede confirmar ejecutando los siguientes comandos en el terminal del MySQL:

```
mysql> show tables;
```

```
+-----+
| Tables_in_Quotes |
+-----+
| gspc              |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select * from gspc;
```

La última instrucción debería imprimir un conjunto grande de registros.

3.3. Definiendo las tareas de predicción.

3.3.1. ¿Qué predecir?

Se desea tener una predicción del comportamiento general de los precios en los siguientes k días, y no capturar solamente un precio particular para un tiempo específico.

Se describirá una variable, calculada por cuotas de datos, que puede ser vista como un indicador (un valor) de la tendencia en los siguientes k días. El valor de este indicador debería estar relacionado con la confianza que se tiene que la meta marginal p será sostenible en los k días siguientes. Es importante destacar que cuando se menciona una variación del $p\%$, significa que ocurre una fluctuación por encima o por debajo del precio actual. La idea es que variaciones positivas lleven a la compra, mientras que variaciones negativas activen la venta de acciones. Este indicador propuesto resume la tendencia hacia un determinado valor, será positivo para tendencias a la alza y negativo para tendencias a la baja de precios.

El precio promedio diario será calculado mediante la siguiente aproximación:

$$\bar{P}_i = \frac{C_i + H_i + L_i}{3}$$

Donde C_i , H_i y L_i son los precios de cierre, valor más alto y valor más bajo para el día i , respectivamente.

Sea V_i el conjunto de las k variaciones porcentuales para el cierre del día de los precios promedios para los siguientes k días (a menudo llamado retorno aritmético):

$$V_i = \left\{ \frac{\bar{P}_{i+j} - C_i}{C_i} \right\}_{j=1}^k$$

La variable indicador es la suma total de las variaciones cuyo valor absoluto es objetivo marginal $p\%$:

$$T_i = \sum_v \{v \in V_i : v > p\% \vee v < -p\%\}$$

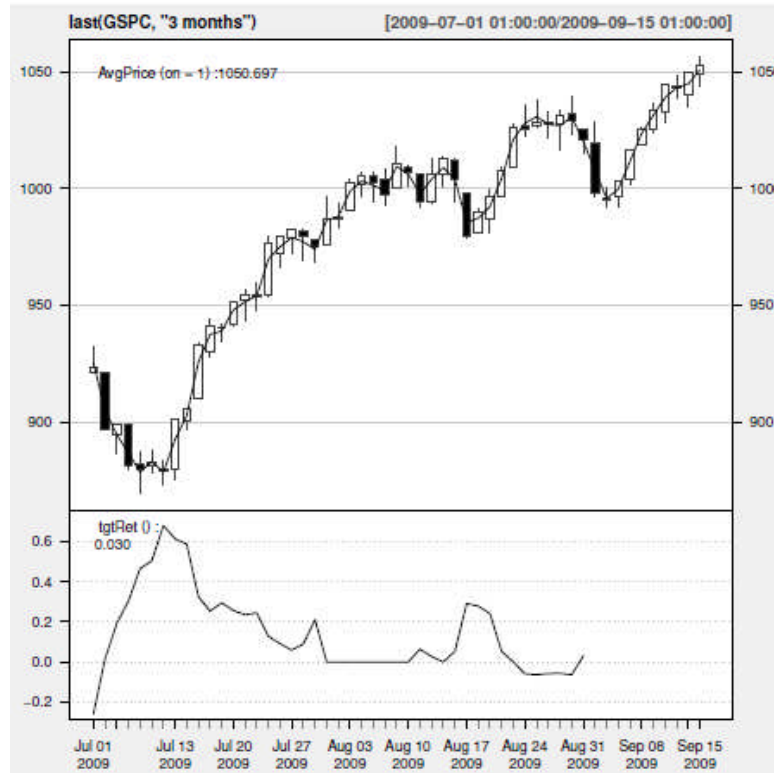
La idea general de la variable T es generar una señal con periodos de k días que tengan varios días con precios diarios promedios claramente acerca de la variación. Valores positivos altos de T significa que hay varios promedios de precios diarios que son más altos que $p\%$. Tal situación son buenos indicios de la oportunidad potencial para comprar acciones, así como buenas expectativas de que el precio seguirá subiendo. Por otra parte, valores altamente negativos de T sugieren la venta de acciones, dado que los precios probablemente disminuyan. Valores alrededor de cero pueden ser causados por periodos con precios “planos” (sin variación) o por conflictos entre variaciones positivas y negativas que se cancelan entre ellos.

La siguiente función implemente este indicador sencillo:

```
> T.ind <- function(quotes, tgt.margin = 0.025, n.days = 10) {
+   v <- apply(HLC(quotes), 1, mean)
+   r <- matrix(NA, ncol = n.days, nrow = NROW(quotes))
+   for (x in 1:n.days) r[, x] <- Next(Delt(v, k = x), x)
+   x <- apply(r, 1, function(x) sum(x[x > tgt.margin | x <
+     -tgt.margin]))
+   if (is.xts(quotes))
+     xts(x, time(quotes))
+   else x
+ }
```

Para tener una mejor idea del comportamiento de este indicador se muestra una figura, la cual fue generada a partir del siguiente código:

```
> candleChart(last(GSPC, "3 months"), theme = "white", TA = NULL)
> avgPrice <- function(p) apply(HLC(p), 1, mean)
> addAvgPrice <- newTA(FUN = avgPrice, col = 1, legend = "AvgPrice")
> addT.ind <- newTA(FUN = T.ind, col = "red", legend = "tgtRet")
> addAvgPrice(on = 1)
> addT.ind()
```

Como se puede observar en la figura el indicador T alcanza valores más altos cuando existe un periodo subsecuente de variaciones positivas. Obviamente, para obtener el valor del indicador para el tiempo i , se necesita los datos para los siguientes 10 días, no estamos diciendo que T anticipa estos movimientos. Esta no es la meta del indicador. La meta es resumir el comportamiento futuro observado de los precios en un valor simple y no predecir este comportamiento.

La interpretación para el tiempo t será “comprar” si el valor de T es más alto que un cierto umbral, y “vender” si dicho valor está por debajo de otro umbral. En otro caso, la señal será no hacer nada (“mantener”). En resumen, se desea predecir la señal correcta para el tiempo t .

3.4. Modelos de predicción.

En esta sección se explorarán algunos modelos que pueden ser utilizados para el manejo de la predicción. Esta selección de modelos se realizó guiada principalmente por el hecho de que estas técnicas son conocidas por su habilidad para el manejo de problemas de regresión no lineales.

3.4.2. Las herramientas de modelado.

3.4.2.1. Redes neuronales artificiales.

Las redes neuronales son utilizadas frecuentemente en las predicciones financieras debido a su habilidad para tratar con problemas no lineales. El paquete **nnet** implementa la red neuronal en R.

La red neuronal artificial está formada por un conjunto de unidades (neuronas) unidas unas a otras. Cada una de las conexiones de las neuronas tiene un peso asociado. Construir una red neuronal artificial consiste en establecer una arquitectura para la red y después utilizar un algoritmo para encontrar los pesos de las conexiones entre las neuronas.

La red organiza sus neuronas en capas. La primera capa contiene las entradas de la red. Las observaciones de entrenamiento son presentadas a la red a través de estas entradas. La capa final contiene las predicciones de la red neuronal para cualquier caso que se presente a las neuronas de entrada. Entre la capa inicial y la final se encuentran las capas ocultas, usualmente se tienen mas de una. Los algoritmos de actualización de los pesos, como el método de “back-propagation”, tratan de obtener los pesos de las conexiones que optimizan un cierto criterio de error, esto es, tratando de asegurar que las salidas de la red se comporte de acuerdo a los casos presentados al modelo.

La red neuronal con una capa oculta puede obtenerse fácilmente en R utilizando la función del paquete **nnet**. La red obtenida por esta función puede ser utilizada para los problemas de clasificación y de regresión, y es aplicable a los problemas de predicción.

Las redes neuronales son conocidas por ser sensibles a diferentes escalas de las variables utilizadas en un problema de predicción. En este contexto, tiene sentido hacer la transformación de los datos antes de introducirlos a la red, para evitar un eventual impacto negativo en el desempeño de la misma. En este caso, se normalizarán los datos con la finalidad de que todas las variables tengan media cero y desviación estándar igual a uno. Esto se puede realizar fácilmente aplicando la siguiente transformación a cada columna del conjunto de datos:

$$y_i = \frac{x_i - \bar{x}}{\sigma_x}$$

Donde \bar{x} es el valor medio de la variable original X, y σ_x es la desviación estándar.

La función **scale()** puede ser utilizada para realizar la transformación de los datos. A continuación se mostrará una simple ilustración de como obtener y usar este tipo de red neuronal artificial en R:


```

> set.seed(1234)
> library(nnet)
> norm.data <- scale(Tdata.train)
> nn <- nnet(Tform, norm.data[1:1000, ], size = 10, decay = 0.01,
+   maxit = 1000, linout = T, trace = F)
> norm.preds <- predict(nn, norm.data[1001:2000, ])
> preds <- unscale(norm.preds, norm.data)

```

Por defecto, la función `nnet()` asigna los pesos iniciales de los enlaces entre los nodos con valores aleatorios entre el intervalo $[-0.5, 0.5]$. Esto significa que dos corridas sucesivas de la función con exactamente los mismo argumentos pueden arrojar diferentes soluciones. Para asegurar que se obtienen los mismo resultados que se presentan a continuación, se añade la **función `set.seed()`** que inicializa el generador de números aleatorios a algún numero de semilla. Esto asegura que se obtendrá exactamente la misma red neuronal que se muestra aquí. En este ejemplo ilustrativo se han usado los primeros 1.000 casos para obtener la red, y para probar el modelo los siguientes 1.000 casos. Después de normalizar los datos de entrenamiento, se llama a la **función `nnet()`** para obtener el modelo. Los primeros dos parámetros son los usuales para cualquier función de modelado en R: la forma funcional del modelo especificado por una fórmula, y las muestras de entrenamiento para obtener el modelo. También se han utilizado algunos de los parámetros de la **función `nnet()`**. El parámetro **`size`** permite especificar cuantos nodos tendrá la capa oculta. No hay receta mágica para utilizar en este caso. Usualmente se intenta con varios valores para observar el comportamiento de la red. Aun así, es razonable asumir que debería ser más pequeño que el número de predictores del problema. El parámetro **`decay`** controla la tasa de actualización de los pesos del algoritmo “back-propagation”. Nuevamente, la técnica de ensayo y error es el mejor amigo aquí. Finalmente el parámetro **`maxit`** controla el número máximo de iteraciones para el proceso de convergencia de los pesos. La instrucción **`linout=T`** dice que la función está manejando un problema de regresión. El comando **`trace=F`** se utiliza para evitar algunos de los resultados de la función en relación con el proceso de optimización.

La función **`predict()`** puede ser utilizada para obtener las predicciones de la red neuronal para un conjunto de datos de prueba. Después de obtener dichas predicciones, se convierten de nuevo los datos a su escala original utilizando la función **`unscale()`** provista por el paquete. Esta función recibe en el primer argumento los valores, y como segundo argumento los objetos con los datos normalizados.

A continuación se evaluarán los resultados de la red neuronal artificial para predecir las señales correctas con el conjunto de datos de prueba. Se transformarán las predicciones numéricas en

señales y se evaluarán utilizando los estadísticos presentados en la sección 3.3.4

```
> sigs.nn <- trading.signals(preds, 0.1, -0.1)
> true.sigs <- trading.signals(Tdata.train[1001:2000, "T.ind.GSPC"],
+   0.1, -0.1)
> sigs.PR(sigs.nn, true.sigs)

      precision    recall
s  0.2101911 0.1885714
b  0.2919255 0.5911950
s+b 0.2651357 0.3802395
```

La función **trading.signals()** transforma las predicciones numéricas en señales, dados los umbrales para comprar y vender, respectivamente. La función **sigs.PR()** obtiene una matriz con las puntuaciones de precisión (precisión) y capacidad para recordar (recall) de los dos tipos de eventos, y en general. Las puntuaciones muestran que el desempeño de la red neuronal no es tan brillante. Bajas puntuaciones en la precisión significa que el modelo arrojó señales incorrectas con bastante frecuencia. Si las señales son utilizadas para el comercio, esto puede llevar a serias pérdidas de dinero.

La red neuronal artificial también puede ser utilizada para tareas de clasificación. Para estos problemas la diferencia principal en términos de la topología de la red, es que en lugar de un solo valor como salida, se tendrán tantas unidades de salida como valores tenga la variable objetivo (algunas veces se conoce como clases de variable). Cada una de estas unidades de salida producirá una probabilidad estimada del respectivo valor de la clase. Esto significa que para cada caso de prueba, una red neuronal puede producir un conjunto de valores de probabilidades, uno por cada valor de clase posible.

El uso de la función **nnet()** para estas tareas es muy similar al uso para problemas de regresión. El siguiente código ilustra esto, utilizando los datos de entrenamiento:

```
> set.seed(1234)
> library(nnet)
> signals <- trading.signals(Tdata.train[, "T.ind.GSPC"], 0.1,
+   -0.1)
> norm.data <- data.frame(signals = signals, scale(Tdata.train[,
+   -1]))
> nn <- nnet(signals ~ ., norm.data[1:1000, ], size = 10, decay = 0.01,
+   maxit = 1000, trace = F)
> preds <- predict(nn, norm.data[1001:2000, ], type = "class")
```

El argumento `type="class"` es usado para obtener una etiqueta de clase sencilla para cada caso de prueba en lugar de un conjunto de probabilidades estimadas. Con las predicciones de la red se

puede calcular la precisión del modelo y su capacidad para recordar, tal como se muestra a continuación:

```
> sigs.PR(preds, norm.data[1001:2000, 1])  
  
      precision    recall  
s    0.2838710 0.2514286  
b    0.3333333 0.2264151  
s+b  0.2775665 0.2185629
```

Los dos resultados, tanto la precisión como la capacidad para recordar son mayores que los obtenidos en la regresión, aun así son valores bajos.

3.4.2.2. Máquinas de soporte vectoriales.

Las máquinas de soporte vectoriales (SVMs) son herramientas de modelado que al igual que las redes neuronales pueden ser aplicadas para tareas de regresión y clasificación. En R ha varias implementaciones de dicha técnica, entre las cuales se puede hacer referencia al **paquete kernlab** con varias funcionalidades disponibles, también la **función svm()** del **paquete e1071**.

La idea básica detrás de los SVMs es la transformación o “mapeo” de los datos originales en uno nuevo espacio multidimensional, donde es posible la aplicación de modelos lineales para obtener un hiper plano de separación, por ejemplo, la separación de clases de un problema, es el caso de la clasificación. El “mapeo” o transformación de los datos originales en un nuevo espacio es llevado a cabo con la ayuda de las funciones denominadas Kernel.

La separación de los hiperplanos en la nueva representación dual se realiza con frecuencia mediante la maximización de un margen de separación entre los casos que pertenecen a clases diferentes ver Figura 3.4. Este es un problema de optimización a menudo resuelto con métodos de programación cuadrática.

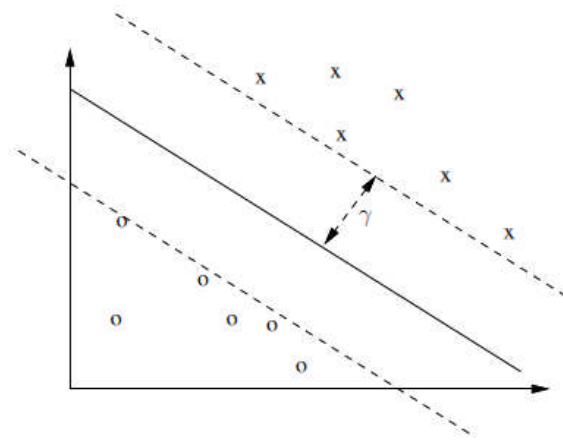


FIGURE 3.4: The margin maximization in SVMs.

A continuación se mostrará ejemplos muy sencillos del uso de este tipo de modelos en R. se comenzará con la tarea de regresión para la cual se utilizará la función provista en el **paquete e1071**:

```
> library(e1071)
> sv <- svm(Tform, Tdata.train[1:1000, ], gamma = 0.001, cost = 100)
> s.preds <- predict(sv, Tdata.train[1001:2000, ])
> sigs.svm <- trading.signals(s.preds, 0.1, -0.1)
> true.sigs <- trading.signals(Tdata.train[1001:2000, "T.ind.GSPC"],
+ 0.1, -0.1)
> sigs.PR(sigs.svm, true.sigs)
```

	precision	recall
s	0.4285714	0.03428571
b	0.3333333	0.01257862
s+b	0.4000000	0.02395210

En este ejemplo se ha utilizado la **función svm()** con la mayoría de los parámetros por defecto, a excepción de los parámetros **gamma** y **costo**. En este contexto, la función utiliza la función radial básica kernel:

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \times \|\mathbf{x} - \mathbf{y}\|^2)$$

El parámetro **costo** indica el costo de la violación del margen. Si desea puede explorar la página de ayuda de la función para aprender con más detalle cada uno de los parámetros de la función.

Como se puede observar, el modelo de SVM alcanza un valor mejor que el obtenido con la red neuronal artificial en términos de precisión, aunque con un valor de reconocimiento (recall) mucho menor.

A continuación se considerará la clasificación, esta vez utilizando el **paquete kernlab**:

```
> library(kernlab)
> data <- cbind(signals = signals, Tdata.train[, -1])
> ksv <- ksvm(signals ~ ., data[1:1000, ], C = 10)
```

Utilizando automáticamente la estimación sigma se tiene:

```
> ks.preds <- predict(ksv, data[1001:2000, ])
> sigs.PR(ks.preds, data[1001:2000, 1])

      precision    recall
s   0.1935484 0.2742857
b   0.2688172 0.1572327
s+b 0.2140762 0.2185629
```

Se ha utilizado el parámetro C de la **función ksvm()** del **paquete kernlab**, para especificar un costo diferente de la violación de las restricciones, la cual por defecto tiene un valor de 1. Aparte de esto se ha utilizado los valores de los parámetros por defecto, para lo referente a la clasificación. Una vez más, los detalles pueden obtenerse en la página de ayuda de la **función ksvm()**.

Los resultados para este SVM no son tan interesantes como los obtenidos para la regresión. Cabe destacar que estos resultados no son los mejores que se podrían obtener con estas técnicas. Estos resultados fueron meramente ilustrativos sobre el uso de estas técnicas de modelado en R.